

[MC-SQLR]: SQL Server Resolution Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft [Open Specification Promise](#) or the [Community Promise](#). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
08/10/2007	0.1	Major	Initial Availability
09/28/2007	0.2	Minor	Updated the technical content.
10/23/2007	0.2.1	Editorial	Revised and edited the technical content.
11/30/2007	0.2.2	Editorial	Revised and edited the technical content.
01/25/2008	0.2.3	Editorial	Revised and edited the technical content.
03/14/2008	0.3	Minor	Updated the technical content.
05/16/2008	0.3.1	Editorial	Revised and edited the technical content.
06/20/2008	0.4	Minor	Updated the technical content.
07/25/2008	0.5	Minor	Updated the technical content.
08/29/2008	0.5.1	Editorial	Revised and edited the technical content.
10/24/2008	0.5.2	Editorial	Revised and edited the technical content.
12/05/2008	0.6	Minor	Updated the technical content.
01/16/2009	0.6.1	Editorial	Revised and edited the technical content.
02/27/2009	1.0	Major	Updated and revised the technical content.
04/10/2009	1.0.1	Editorial	Revised and edited the technical content.
05/22/2009	1.1	Minor	Updated the technical content.
07/02/2009	1.1.1	Editorial	Revised and edited the technical content.
08/14/2009	1.1.2	Editorial	Revised and edited the technical content.
09/25/2009	1.2	Minor	Updated the technical content.
11/06/2009	1.3	Minor	Updated the technical content.
12/18/2009	1.3.1	Editorial	Revised and edited the technical content.
01/29/2010	1.4	Minor	Updated the technical content.
03/12/2010	1.4.1	Editorial	Revised and edited the technical content.
04/23/2010	1.5	Minor	Updated the technical content.
06/04/2010	1.6	Minor	Updated the technical content.
07/16/2010	1.6	No change	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
08/27/2010	1.6	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	2.0	Major	Significantly changed the technical content.
11/19/2010	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	3.0	Major	Significantly changed the technical content.
02/11/2011	4.0	Major	Significantly changed the technical content.
03/25/2011	5.0	Major	Significantly changed the technical content.
05/06/2011	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	5.1	Minor	Clarified the meaning of the technical content.
09/23/2011	5.2	Minor	Clarified the meaning of the technical content.
12/16/2011	6.0	Major	Significantly changed the technical content.
03/30/2012	6.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	7.0	Major	Significantly changed the technical content.
10/25/2012	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/31/2013	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
08/08/2013	8.0	Major	Significantly changed the technical content.
11/14/2013	8.0	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction	6
1.1 Glossary	6
1.2 References	7
1.2.1 Normative References	7
1.2.2 Informative References	7
1.3 Overview	8
1.4 Relationship to Other Protocols	9
1.5 Prerequisites/Preconditions	9
1.6 Applicability Statement	10
1.7 Versioning and Capability Negotiation	10
1.8 Vendor-Extensible Fields	10
1.9 Standards Assignments	10
2 Messages	11
2.1 Transport	11
2.2 Message Syntax	11
2.2.1 CLNT_BCAST_EX	11
2.2.2 CLNT_UCAST_EX	11
2.2.3 CLNT_UCAST_INST	12
2.2.4 CLNT_UCAST_DAC	12
2.2.5 SVR_RESP	13
2.2.6 SVR_RESP (DAC)	15
3 Protocol Details	17
3.1 Server Details	17
3.1.1 Abstract Data Model	17
3.1.2 Timers	17
3.1.3 Initialization	17
3.1.4 Higher-Layer Triggered Events	17
3.1.5 Message Processing Events and Sequencing Rules	17
3.1.5.1 Initial State	18
3.1.5.2 Waiting For Request From Client	18
3.1.6 Timer Events	18
3.1.7 Other Local Events	18
3.2 Client Details	18
3.2.1 Abstract Data Model	19
3.2.2 Timers	19
3.2.3 Initialization	20
3.2.4 Higher-Layer Triggered Events	20
3.2.5 Message Processing Events and Sequencing Rules	20
3.2.5.1 Begin	20
3.2.5.2 Client Waits For Response From Server	20
3.2.5.3 Client Waits For Response From Server(s)	20
3.2.5.4 Waiting Completed	21
3.2.5.5 End	21
3.2.6 Timer Events	21
3.2.7 Other Local Events	21
4 Protocol Examples	22
4.1 CLNT_UCAST_EX	22

4.2	CLNT_UCAST_INST	22
4.3	CLNT_UCAST_DAC	23
5	Security	24
5.1	Security Considerations for Implementers	24
5.2	Index of Security Parameters	24
6	Appendix A: Product Behavior	25
7	Change Tracking.....	27
8	Index	28

1 Introduction

The SQL Server Resolution Protocol is an application-layer request/response protocol that facilitates connectivity to a database **server**. This protocol provides for the following:

- Communication **endpoint** information; for example, the TCP port for connecting to a particular **instance** of the database server on a machine.
- Database instance enumeration.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

**broadcast
endpoint
Internet Protocol version 4 (IPv4)
Internet Protocol version 6 (IPv6)
little-endian
named pipe
NetBIOS name
Transmission Control Protocol (TCP)
Virtual Interface Architecture (VIA)**

The following terms are specific to this document:

client: A program that establishes connections for the purpose of sending requests.

data store: A repository for data.

database server discovery service: A service that allows applications to discover the existence of database **instances**.

dedicated administrator connection (DAC): A special TCP **endpoint** that was introduced in SQL Server 2005. DAC provides a special diagnostic connection for administrators when standard connections to the server are not possible.

instance: A copy of the database server that is running on a machine. Multiple instances may reside on a single machine.

multicast: A data transmission to multiple specific recipients at the same time.

query: A character string expression that is sent to a **data store**. The query contains a set of operations that request data from the **data store**.

server: An application program that accepts connections to service requests by sending back responses. Any program may be capable of being both a **client** and a server. Use of these terms refers only to the role that is performed by the program for a particular connection, rather than to the program's capabilities in general.

unicast: A data transmission to a single specific recipient.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [\[RFC2119\]](#). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

A reference marked "(Archived)" means that the reference document was either retired and is no longer being maintained or was replaced with a new document that provides current implementation details. We archive our documents online [\[Windows Protocol\]](#).

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, <http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624>, as an additional source.

[MS-UCODEREF] Microsoft Corporation, "[Windows Protocols Unicode Reference](#)".

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980, <http://www.ietf.org/rfc/rfc768.txt>

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981, <http://www.ietf.org/rfc/rfc791.txt>

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981, <http://www.ietf.org/rfc/rfc0793.txt>

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/rfc/rfc2119.txt>

[RFC2460] Deering, S., and Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998, <http://www.ietf.org/rfc/rfc2460.txt>

[RFC4234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, <http://www.ietf.org/rfc/rfc4234.txt>

[VIA2002] Cameron, D., and Regnier, G., "The Virtual Interface Architecture", Intel Press, 2002, ISBN:0971288704.

If you have any trouble finding [VIA2002], please check [here](#).

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)".

[MSDN-CS] Microsoft Corporation, "Character Sets", <http://msdn.microsoft.com/en-us/library/dd317743.aspx>

[MSDN-DAC] Microsoft Corporation, "Using a Dedicated Administrator Connection", <http://msdn.microsoft.com/en-us/library/ms189595.aspx>

[MSDN-NamedPipes] Microsoft Corporation, "Creating a Valid Connection String Using Named Pipes", [http://msdn.microsoft.com/en-us/library/ms189307\(SQL.100\).aspx](http://msdn.microsoft.com/en-us/library/ms189307(SQL.100).aspx)

[PIPE] Microsoft Corporation, "Named Pipes", <http://msdn.microsoft.com/en-us/library/aa365590.aspx>

1.3 Overview

The SQL Server Resolution Protocol is a simple application-level protocol that is used for the transfer of requests and responses between **clients** and **database server discovery services**.^{<1>} In such a system, the client either (i) sends a single request to a specific machine and expects a single response, or (ii) **broadcasts** or **multicasts** a request to the network and expects zero or more responses from different discovery services on the network. The first case is used for the purpose of determining the communication endpoint information of a particular database instance, whereas the second case is used for enumeration of database instances in the network and to obtain the endpoint information of each instance.

The SQL Server Resolution Protocol does not include any facilities for authentication, protection of data, or reliability. The SQL Server Resolution Protocol is always implemented on top of the UDP Transport Protocol [RFC768].

In the case of endpoint determination for a single instance, the following diagram depicts a typical flow of communication.

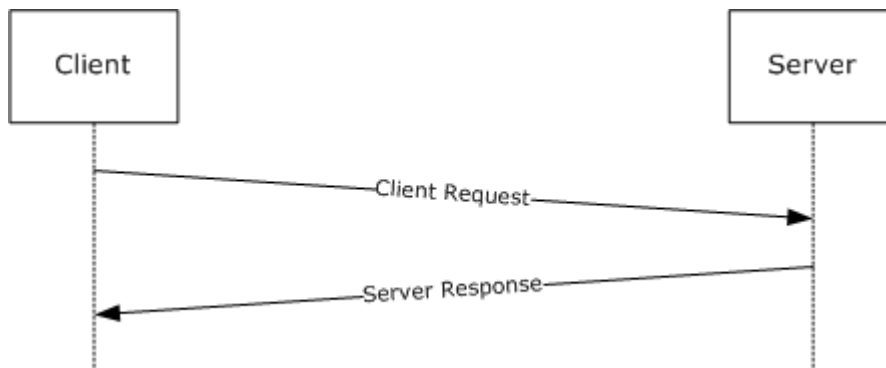


Figure 1: Communication flow for single-instance endpoint discovery

Conversely, in the case of a broadcast/multicast request, the following diagram applies.

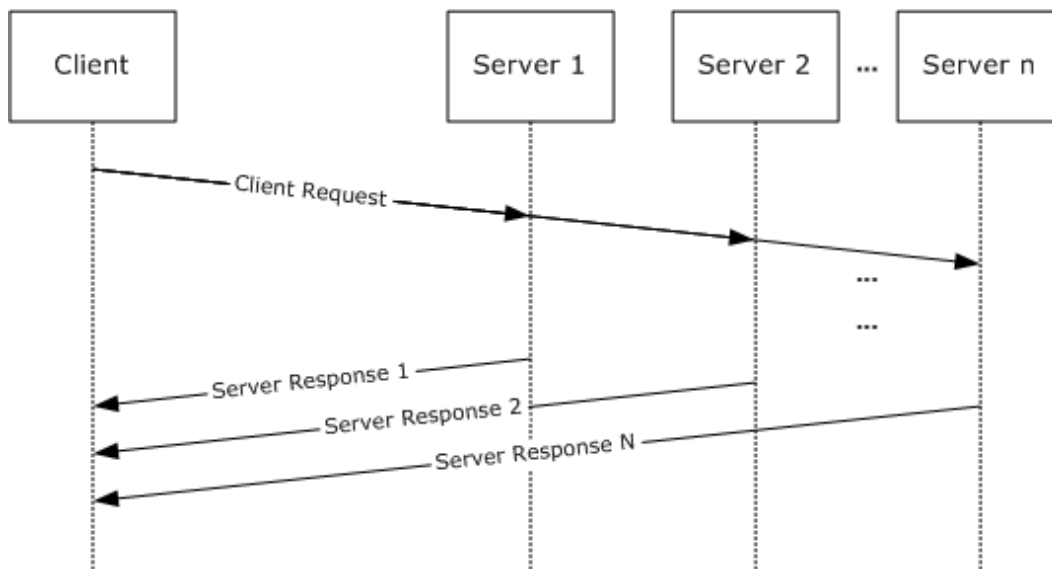


Figure 2: Communication flow for multiple-instance endpoint discovery

In the case of a broadcast or multicast request, the client does not necessarily know the number of responses that it can expect. As a result, it is reasonable for the client to enforce a time limitation during which it waits for responses. Because some servers may not respond quickly enough or may not receive the request (highly dependent on network topology), the broadcast/multicast request for multiple-instance endpoint information should be considered nondeterministic.

1.4 Relationship to Other Protocols

The SQL Server Resolution Protocol (SSRP) depends on the UDP Transport Protocol to communicate with the database server machine or to broadcast/multicast its request to the network. The types of addresses used may differ based on the underlying IP protocol version as described in [2.1](#). For details about **IPv4**, see [\[RFC791\]](#). For details about **IPv6**, see [\[RFC2460\]](#).

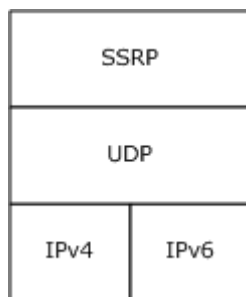


Figure 3: Protocol relationship

1.5 Prerequisites/Preconditions

Unprohibited access to UDP port 1434 is required.

1.6 Applicability Statement

The SQL Server Resolution Protocol is appropriate for use to facilitate retrieval of database endpoint information or for database instance enumeration in all scenarios where network or local connectivity is available.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- Supported transports: This protocol is implemented on top of UDP, as discussed in section [2.1](#).
- Protocol versions: The SQL Server Resolution Protocol supports the following explicit dialect: "SSRP 1.0", as defined in section [2.2](#).
- Security and authentication methods: The SQL Server Resolution Protocol does not provide or support any security or authentication methods.
- Localization: Localization-dependent protocol behavior is specified in sections [2.2](#) and [3.2.5](#).
- Capability negotiation: The SQL Server Resolution Protocol does not support negotiation of the dialect to use. Instead, an implementation must be configured with the dialect to use, as described later in this section.

No version or capability negotiation is supported by the SQL Server Resolution Protocol. For example, the client sends a request message to the server with the expectation that the server will understand the message and send back a response. If the server does not understand the message, the server ignores the request and does not send a response back to the client.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

Parameter	Value	Reference
Microsoft-SQL-Monitor (ms-sql-m)	1434/UDP	http://www.iana.com/assignments/port-numbers

The client always sends its request to UDP port 1434 of the server or servers.

2 Messages

2.1 Transport

The SQL Server Resolution Protocol uses port 1434 of the UDP Protocol for message transport. When using the UDP Protocol over the IPv4 Protocol, the SQL Server Resolution Protocol uses a **unicast** or broadcast address, depending on the message type, as specified in section 2.2. When using the UDP Protocol over the IPv6 Protocol, the SQL Server Resolution Protocol uses a unicast or multicast address, depending on the message type, as specified in section 2.2.

2.2 Message Syntax

All integer fields are represented in **little-endian** format. All text strings are represented as a multibyte character set (MBCS) string [\[MS-UCODEREF\]](#) on the current system code page of the server and the client. (The system code page on the client and the system code page on the server are assumed to be common.) They are not case-sensitive. See [\[MSDN-CS\]](#) for more information.

The valid requests from the client are as follows:

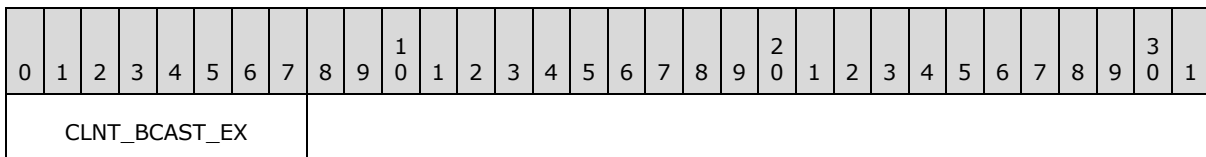
- [CLNT_BCAST_EX](#)
- [CLNT_UCAST_EX](#)
- [CLNT_UCAST_INST](#)
- [CLNT_UCAST_DAC](#)

The response from the server is always an [SVR_RESP](#) message. The response contains a string data field that is parsed by the client to extract the requested information. The contents of this string are not case-sensitive.

These messages are explained in more detail in the following sections.

2.2.1 CLNT_BCAST_EX

The CLNT_BCAST_EX packet is a broadcast or multicast request that is generated by clients that are trying to identify the list of database instances on the network and their network protocol connection information.



CLNT_BCAST_EX (1 byte): A single byte whose value MUST be 0x02.

2.2.2 CLNT_UCAST_EX

The CLNT_UCAST_EX packet is a unicast request that is generated by clients that are trying to identify the list of database instances and their network protocol connection information installed on a single machine. The client generates a UDP packet with a single byte, as shown in the following diagram.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31					
CLNT_UCAST_EX																																				

CLNT_UCAST_EX (1 byte): A single byte whose value MUST be 0x03.

2.2.3 CLNT_UCAST_INST

The CLNT_UCAST_INST packet is a request for information related to a specific instance. The structure of the request is as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31					
CLNT_UCAST_INST										INSTANCENAME (variable)																										
...																																				

CLNT_UCAST_INST (1 byte): A single byte whose value MUST be 0x04.

INSTANCENAME (variable): A variable-length null-terminated multibyte character set (MBCS) string that does not need to be byte-aligned. It MUST be no greater than 32 bytes in length, not including the null terminator.

Note

- INSTANCENAME corresponds to the name of the database instance for which the information is requested.

2.2.4 CLNT_UCAST_DAC

The CLNT_UCAST_DAC packet request is used to determine the [TCP \[RFC793\]](#) port on which the SQL Server **dedicated administrator connection (DAC)** [\[MSDN-DAC\]](#) endpoint is listening.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CLNT_UCAST_DAC										PROTOCOLVERSION										INSTANCENAME (variable)											
...																															

CLNT_UCAST_DAC (1 byte): A single byte whose value MUST be 0x0F.

PROTOCOLVERSION (1 byte): A single byte whose value MUST be 0x01.

INSTANCENAME (variable): A variable-length null-terminated multibyte character set (MBCS) string that does not need to be byte-aligned. It MUST be no greater than 32 bytes in length, not including the null terminator.

2.2.5 SVR_RESP

The server responds to all client requests with an SVR_RESP. There is a slight variation in the message format for a response to a [CLNT_UCAST_DAC \(section 2.2.4\)](#) request, as described in the next section.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1		
SVR_RESP										RESP_SIZE										RESP_DATA (variable)													
...																																	

SVR_RESP (1 byte): A single byte whose value MUST be 0x05.

RESP_SIZE (2 bytes): This unsigned integer specifies the length, in bytes, of subsequent response data **RESP_DATA**.

RESP_DATA (variable): A variable-length MBCS string that does not need to be byte-aligned. The maximum size of **RESP_DATA** MUST be 1,024 bytes if the server is responding to a [CLNT_UCAST_INST \(section 2.2.3\)](#) request. If the server is responding to a [CLNT_BCAST_EX \(section 2.2.1\)](#) or a [CLNT_UCAST_EX \(section 2.2.2\)](#) request, the maximum size of **RESP_DATA** MUST be 65,535 bytes.

Note

- The content of the **RESP_DATA** string corresponds to the type of request received. For example, a CLNT_UCAST_EX request is answered with a CLNT_UCAST_EX response string. The formal syntax of the message is provided in Augmented Backus-Naur Form (ABNF), as specified in [\[RFC4234\]](#).
- All responses to client requests contain information about one or more of the following transports:
 - TCP [\[RFC793\]](#)
 - **Named Pipes** in message mode [\[PIPE\]](#). See [\[MSDN-NamedPipes\]](#) for additional information related to Microsoft-specific implementations.
 - A reliable transport over the **Virtual Interface Architecture (VIA)** interface [\[VIA2002\]](#).

```
RESP_DATA = CLNT_BCAST_EX_RESPONSE / CLNT_UCAST_EX_RESPONSE / CLNT_UCAST_INST_RESPONSE
```

```
CLNT_BCAST_EX_RESPONSE = CLNT_UCAST_EX_RESPONSE
```

```
CLNT_UCAST_EX_RESPONSE = *[CLNT_UCAST_INST_RESPONSE]
```

```
CLNT_UCAST_INST_RESPONSE = "ServerName" SEMICOLON SERVERNAME SEMICOLON
"InstanceName" SEMICOLON INSTANCENAME SEMICOLON "IsClustered"
SEMICOLON YES_OR_NO SEMICOLON "Version" SEMICOLON VERSION_STRING
[NP_INFO] [TCP_INFO] [VIA_INFO] [RPC_INFO] [SPX_INFO] [ADSP_INFO]
[BV_INFO] SEMICOLON SEMICOLON ; see Note 1
```

Name	Value
SEMICOLON	SEMICOLON = " ; "
SERVERNAME	The name of the server. The SERVERNAME MUST be no greater than 255 bytes.
INSTANCENAME	A text string that represents the name of the server instance being described. The INSTANCENAME MUST be no greater than 255 bytes but SHOULD be no greater than 16 MBCS characters.
YES_OR_NO	YES_OR_NO= "Yes" / "No"
VERSION_STRING	A text string that conveys the version of the server instance. The VERSION_STRING MUST be no greater than 16 bytes. VERSION_STRING MUST NOT be empty and MUST appear as follows: VERSION_STRING=1*[0-9"."]
NP_INFO	NP_INFO=SEMICOLON "np" SEMICOLON NP_PARAMETERS
NP_PARAMETERS	NP_PARAMETERS=PIPENAME
PIPENAME	A text string that represents the pipe name [PIPE] .
TCP_INFO	TCP_INFO=SEMICOLON "tcp" SEMICOLON TCP_PARAMETERS
TCP_PARAMETERS	TCP_PARAMETERS = TCP_PORT
TCP_PORT	A text string that represents the decimal value of the TCP port that is used to connect to the requested server instance. TCP_PORT SHOULD be a valid TCP port as specified in [RFC793] ; see Note 2.
VIA_INFO	VIA_INFO=SEMICOLON "via" SEMICOLON VIA_PARAMETERS; see Note 3.
VIA_PARAMETERS	VIA_PARAMETERS=NETBIOS VIALISTENINFO
VIALISTENINFO	VIALISTENINFO =1*[" , " VIANIC ":" VIAPORT].
VIANIC	A text string that represents the VIA network interface card (NIC) identifier. VIANIC SHOULD be a valid VIA Adapter NIC number [VIA2002] .
VIAPORT	A text string that represents the decimal value of the VIA NIC's port. VIAPORT SHOULD be a valid VIA Adapter port number [VIA2002] .
NETBIOS	A text string that MUST be no greater than 15 bytes and that represents the NetBIOS name of a machine where the server resides.
RPC_INFO	SEMICOLON "rpc" SEMICOLON RPC_PARAMETERS
RPC_PARAMETERS	RPC_PARAMETERS=COMPUTERNAME
COMPUTERNAME	The name of the computer to connect to. SHOULD be no more than 127 MBCS characters.
SPX_INFO	SPX_INFO=SEMICOLON "spx" SEMICOLON SPX_PARAMETERS
SPX_PARAMETERS	SPX_PARAMETERS=SERVICENAME
SERVICENAME	The SPX service name of the server. MUST NOT be greater than 1,024 bytes and SHOULD be no more than 127 MBCS characters.

Name	Value
ADSP_INFO	ADSP_INFO=SEMICOLON "adsp" SEMICOLON ADSP_PARAMETERS
ADSP_PARAMETERS	ADSP_PARAMETERS=ADSP_OBJECTNAME
ADSP_OBJECTNAME	The AppleTalk service object name. SHOULD be no more than 127 MBCS characters.
BV_INFO	BV_INFO=SEMICOLON "bv" SEMICOLON ITEMNAME SEMICOLON GROUPNAME SEMICOLON BV_PARAMETERS
BV_PARAMETERS	BV_PARAMETERS=ITEMNAME SEMICOLON GROUPNAME SEMICOLON ORGNAME
ITEMNAME	The Banyan VINES item name. SHOULD be no more than 127 MBCS characters.
GROUPNAME	The Banyan VINES group name. SHOULD be no more than 127 MBCS characters.
ORGNAME	The Banyan VINES organization name. SHOULD be no more than 127 MBCS characters.

Note

1. NP_INFO, TCP_INFO, VIA_INFO, RPC_INFO, SPX_INFO, ADSP_INFO, and BV_INFO may be listed in any order and MUST be listed no more than once; multiple protocols can be sent in the response. The TCP_PORT in TCP_INFO MUST be in the range defined by [\[RFC793\].<2>](#)
2. VIA_INFO SHOULD cumulatively be no greater than 128 bytes.
3. The size of the SQL Server Resolution Protocol response MUST be no greater than 1,024 bytes per instance.

2.2.6 SVR_RESP (DAC)

The format of the SVR_RESP is different for a [CLNT_UCAST_DAC](#) request only.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SVR_RESP										RESP_SIZE										PROTOCOLVERSION											
TCP_DAC_PORT																															

SVR_RESP (1 byte): A single byte whose value MUST be 0x05.

RESP_SIZE (2 bytes): This unsigned integer specifies the length of the entire response packet, in bytes. The value of this field MUST be 0x06.

PROTOCOLVERSION (1 byte): A single byte whose value MUST be 0x01.

TCP_DAC_PORT (2 bytes): The value of the TCP port number that is used for DAC.

Note

- All responses to client requests contain information about the following transport:

- TCP [\[RFC793\]](#)

3 Protocol Details

This section describes the important elements of the client software and the server software necessary to support the SQL Server Resolution Protocol.

As described in section 1.3, the SQL Server Resolution Protocol is an application-level protocol that is used to **query** information regarding database instances on one or more servers. The protocol defines a limited set of messages through which the client may make a request to the server or servers. The SQL Server Resolution Protocol involves a single request and a single response from one or more servers. The response contains instance-specific values provided by the higher layer.

3.1 Server Details

The following state machine diagram describes the server side of the SQL Server Resolution Protocol.

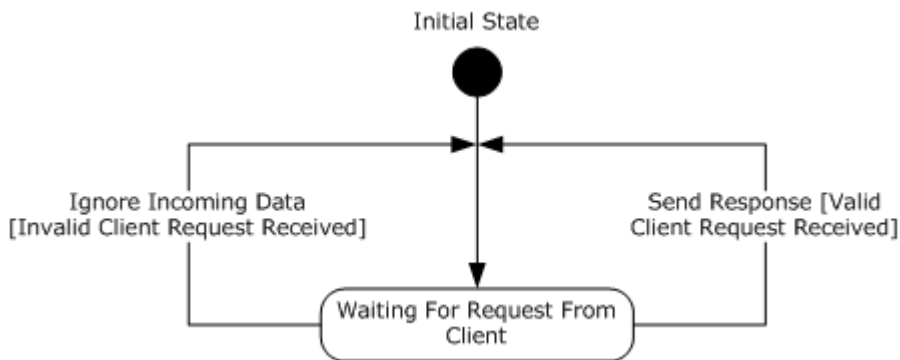


Figure 4: SQL Server Resolution Protocol server state machine

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

The SQL Server Resolution Protocol does not perform any initialization on the server side. A UDP socket that is listening on port 1434 is assumed to have been created by the higher layer.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

Because the SQL Server Resolution Protocol provides a single response per server for each client request, no sequencing issues occur with this protocol.

3.1.5.1 Initial State

In the "Initial State", the initialization found in section [3.1.3](#) is assumed to have taken place. Upon success, the server MUST immediately enter the "Waiting For Request From Client" state.

3.1.5.2 Waiting For Request From Client

In the "Waiting For Request From Client" state, the server listens on UDP port 1434 for an incoming request. If the request is valid and understood, the server immediately sends an [SVR_RESP](#) response back to the client. The data content of the response depends on the request type.

- For [CLNT_BCAST_EX](#) and [CLNT_UCAST_EX](#), the server returns information about all available instances. The information about all available instances is provided by the higher layer.
- For [CLNT_UCAST_INST](#), the server returns information about the specified instance only (if available). The information about the specified instance is provided by the higher layer.
- For [CLNT_UCAST_DAC](#), the server returns information about the dedicated administrator connection (DAC) only.

The response SHOULD include information for a particular protocol as long as the aggregate information for the instance fits within the 1,024 bytes limit. If the information for a protocol would cause the total information for all protocols to exceed 1,024 bytes - for example, trying to add a 500-byte pipe name when 800 bytes of response data have already been collected-the information for this protocol SHOULD not be sent. The information for the next protocol (if any) SHOULD be included in the response (assuming that it does not cause the response to exceed the 1,024 bytes limit). Furthermore, the server SHOULD NOT include a protocol and its information if no valid information is available. For example, if the TCP port is invalid, TCP would not be included in the response. SQLR SHOULD NOT verify the length or content of the **PIPENAME** field, which is provided by the higher layer. It is the upper layer's responsibility to ensure that **PIPENAME** conforms to the specification of a valid pipe name [\[PIPE\]](#).

If the request is received on an IPv4 socket, the response provides the instance's port that is associated with an IPv4 address, and likewise for IPv6.

If the request is not valid, not understood, or if there is no instance for which it can send back endpoint information, the server MUST ignore the request. The server MUST then enter the "Waiting For Request From Client" state.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Client Details

The following state machine diagram describes the client side of the SQL Server Resolution Protocol.

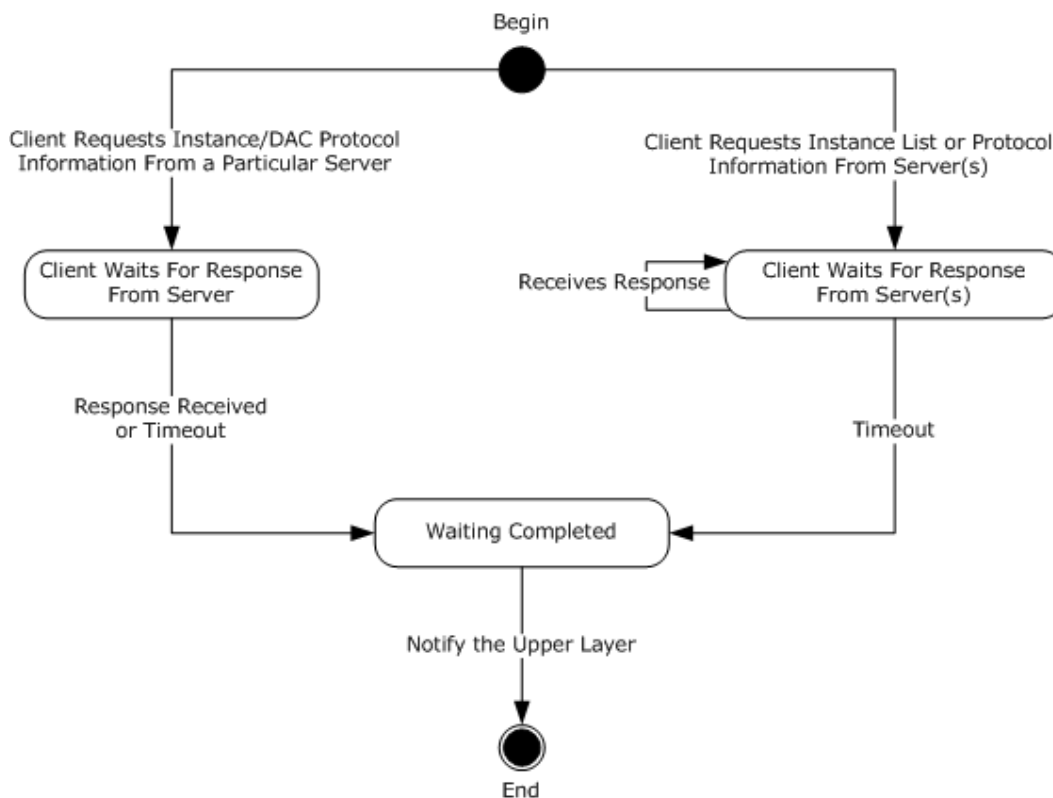


Figure 5: SQL Server Resolution Protocol client state machine

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model provided that their external behavior is consistent with that described in this document.

A SQL Server Resolution Protocol client does not need to maintain any state data except for the knowledge of the request sent to the server.

3.2.2 Timers

The SQL Server Resolution Protocol client **MUST** implement a timer for the amount of time to wait for an [SVR_RESP](#) message from the server when a [CLNT_UCAST_INST](#) or [CLNT_UCAST_DAC](#) request is sent. The timer mechanism that is used is implementation-specific but **SHOULD** [<3>](#) have a time-out value of 1 second.

The SQL Server Resolution Protocol client **MUST** implement a timer for the amount of time to wait for SVR_RESP messages from servers in the network after a [CLNT_UCAST_EX](#) or [CLNT_BCAST_EX](#) request is sent. The timer mechanism that is used is implementation-specific. [<4>](#)

3.2.3 Initialization

The SQL Server Resolution Protocol does not perform any initialization on the client side. A UDP socket is assumed to be created prior to requesting that a SQL Server Resolution Protocol request be sent.

3.2.4 Higher-Layer Triggered Events

The SQL Server Resolution Protocol client implementation **MUST** support the following event from the higher layer:

- Send client request to server or servers. The type of request dictates if the message is sent to a specific machine or broadcast/multicast to the network. The higher layer must have already created a UDP socket prior to triggering the SQL Server Resolution Protocol client to send a request message. After the message is sent, the timer **MUST** be started.

The higher layer is responsible for closing the UDP socket after the response is received or a time-out situation has occurred.

3.2.5 Message Processing Events and Sequencing Rules

Because the SQL Server Resolution Protocol provides a single response per server for each client request, no sequencing issues occur with this protocol.

3.2.5.1 Begin

In the "Begin" state, the client awaits a request from the higher layer. After a request from the higher layer is made, the client sends a request to the server or servers. The request type determines what state the client enters next.

If the client sends a [CLNT_UCAST_INST](#) or [CLNT_UCAST_DAC](#) request to the server, the client **MUST** then enter the "Client Waits For Response From Server" state. If the client sends a [CLNT_UCAST_EX](#) or [CLNT_BCAST_EX](#) request to a server or servers, the client **MUST** then enter the "Client Waits For Response From Server(s)" state.

3.2.5.2 Client Waits For Response From Server

In the "Client Waits For Response From Server" state, the client waits either for a time-out to occur or for the results of a request to return. As soon as either occurs, the client **MUST** enter the "Waiting Completed" state.

The details of the timer are outlined in section [3.2.2](#).

3.2.5.3 Client Waits For Response From Server(s)

In the "Client Waits For Response From Server(s)" state, the client waits for responses up until the time-out expires. If the client receives an invalid message, it **MUST** ignore the message and continue listening for additional responses until the time-out period elapses. The client then **MUST** enter the "Waiting Completed" state.

For purposes of this section, invalid messages are defined as messages that do not follow the prescribed message format that is outlined in section [2](#) or defined as an unexpected [SVR_RESP](#) messages type.

The details of the timer are outlined in section [3.2.2](#).

3.2.5.4 Waiting Completed

The client's actions upon entering the "Waiting Completed" state are determined by the client message type to which the server is responding.

[CLNT_UCAST_DAC](#): The client MUST notify the higher layer of the valid and properly formatted [SVR_RESP](#) (DAC) messages or notify the higher layer if it received an invalid message. After this, the client MUST enter the "End" state.

[CLNT_BCAST_EX](#): The client MUST notify the higher layer of the valid and properly formatted [SVR_RESP](#) messages. The client SHOULD buffer all responses until the timer has timed out. It MUST then pass the information to the higher layer. The client MUST ignore the invalid messages and does not notify the higher layer regarding these messages. After this, the client MUST enter the "End" state.

[CLNT_UCAST_EX](#): The client MUST notify the higher layer of the valid and properly formatted [SVR_RESP](#) messages or notify the higher layer if it received an invalid message. After this, the client MUST enter the "End" state. Although the server's maximum [RESP_DATA](#) size for a [SVR_RESP](#) message is 65,535 bytes, the client MAY consider a [SVR_RESP](#) message improperly formatted if the [RESP_DATA](#) field exceeds a value set by the client that is smaller than 65,535 bytes. <5>

[CLNT_UCAST_INST](#): The client MUST notify the higher layer of the valid and properly formatted [SVR_RESP](#) messages or notify the higher layer if it received an invalid message. After this, the client MUST enter the "End" state. A [SRV_RESP](#) message SHOULD NOT be considered properly formatted if the cumulative length of the parameters of any transport protocol included in the response is more than 255 bytes (in other words, a [SRV_RESP](#) message SHOULD NOT be considered properly formatted if it contains an [NP_PARAMETERS](#), [TCP_PARAMETERS](#), [VIA_PARAMETERS](#), [RPC_PARAMETERS](#), [SPX_PARAMETERS](#), [ADSP_PARAMETERS](#), or [BV_PARAMETERS](#) token whose length is more than 255 bytes).

3.2.5.5 End

The client has completed the request.

3.2.6 Timer Events

When the timer for the response to a broadcast or multicast request expires, the client MUST enter the "Waiting Completed" state.

3.2.7 Other Local Events

None.

4 Protocol Examples

The following are examples of the binary representation of various client requests and the responses from the server.

4.1 CLNT_UCAST_EX

Request:

```
03
```

Response:

```
05 47 01 53 65 72 76 65 72 4e 61 6d 65 3b 49 4c | .G.ServerName;IL
53 55 4e 47 31 3b 49 6e 73 74 61 6e 63 65 4e 61 | SUNG1;InstanceNa
6d 65 3b 59 55 4b 4f 4e 53 54 44 3b 49 73 43 6c | me;YUKONSTD;IsCl
75 73 74 65 72 65 64 3b 4e 6f 3b 56 65 72 73 69 | ustered;No;Versi
6f 6e 3b 39 2e 30 30 2e 31 33 39 39 2e 30 36 3b | on;9.00.1399.06;
74 63 70 3b 35 37 31 33 37 3b 3b 53 65 72 76 65 | tcp;57137;;Serve
72 4e 61 6d 65 3b 49 4c 53 55 4e 47 31 3b 49 6e | rName;ILSUNG1;In
73 74 61 6e 63 65 4e 61 6d 65 3b 59 55 4b 4f 4e | stanceName;YUKON
44 45 56 3b 49 73 43 6c 75 73 74 65 72 65 64 3b | DEV;IsClustered;
4e 6f 3b 56 65 72 73 69 6f 6e 3b 39 2e 30 30 2e | No;Version;9.00.
31 33 39 39 2e 30 36 3b 6e 70 3b 5c 5c 49 4c 53 | 1399.06;np;\\ILS
55 4e 47 31 5c 70 69 70 65 5c 4d 53 53 51 4c 24 | UNG1\pipe\MSSQL$
59 55 4b 4f 4e 44 45 56 5c 73 71 6c 5c 71 75 65 | YUKONDEV\sql\que
72 79 3b 3b 53 65 72 76 65 72 4e 61 6d 65 3b 49 | ry;;ServerName;I
4c 53 55 4e 47 31 3b 49 6e 73 74 61 6e 63 65 4e | LSUNG1;InstanceN
61 6d 65 3b 4d 53 53 51 4c 53 45 52 56 45 52 3b | ame;MSSQLSERVER;
49 73 43 6c 75 73 74 65 72 65 64 3b 4e 6f 3b 56 | IsClustered;No;V
65 72 73 69 6f 6e 3b 39 2e 30 30 2e 31 33 39 39 | ersion;9.00.1399
2e 30 36 3b 74 63 70 3b 31 34 33 33 3b 6e 70 3b | .06;tcp;1433;np;
5c 5c 49 4c 53 55 4e 47 31 5c 70 69 70 65 5c 73 | \\ILSUNG1\pipe\s
71 6c 5c 71 75 65 72 79 3b 3b | ql\query;;
```

The response conveys the instance information for 3 instances named "YUKONSTD", "YUKONDEV", and "MSSQLSERVER".

4.2 CLNT_UCAST_INST

Request:

```
04 59 55 4b 4f 4e 53 54 44 00 | .YUKONSTD
```

The request is for information for an instance named "YUKONSTD".

Response:

```
05 58 00 53 65 72 76 65 72 4e 61 6d 65 3b 49 4c | .X.ServerName;IL
53 55 4e 47 31 3b 49 6e 73 74 61 6e 63 65 4e 61 | SUNG1;InstanceNa
6d 65 3b 59 55 4b 4f 4e 53 54 44 3b 49 73 43 6c | me;YUKONSTD;IsCl
75 73 74 65 72 65 64 3b 4e 6f 3b 56 65 72 73 69 | ustered;No;Versi
```

```
6f 6e 3b 39 2e 30 30 2e 31 33 39 39 2e 30 36 3b | on;9.00.1399.06;  
74 63 70 3b 35 37 31 33 37 3b 3b | tcp;57137;;
```

4.3 CLNT_UCAST_DAC

Request:

```
0f 01 59 55 4b 4f 4e 53 54 44 00 | ..YUKONSTD
```

The request is for the DAC port of an instance named "YUKONSTD".

Response:

```
05 06 00 01 32 df | ....2
```

The port number is 0xDF32 or 57138.

5 Security

5.1 Security Considerations for Implementers

No security considerations are associated with the SQL Server Resolution Protocol.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft SQL Server 2000
- Windows XP operating system
- Windows Server 2003 operating system
- Microsoft SQL Server 2005
- Windows Vista operating system
- Windows Server 2008 operating system
- Microsoft SQL Server 2008
- Windows 7 operating system
- Windows Server 2008 R2 operating system
- Microsoft SQL Server 2008 R2
- Microsoft SQL Server 2012
- Windows Server 2012 operating system
- Windows 8 operating system
- Windows 8.1 operating system
- Windows Server 2012 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

[<1> Section 1.3:](#) SQL Server Browser is an example of a database server discovery service in the Windows environment.

[<2> Section 2.2.5:](#) The information for an instance of SQL Server can include the RPC_INFO, SPX_INFO, ADSP_INFO, and BV_INFO tokens only if the version of the SQL Server instance is SQL Server 2000.

SQL Server 2012 Browser does not support sending information about instances of SQL Server 2000, and will never send these tokens.

[<3> Section 3.2.2:](#) Windows implements the timers for these two messages as follows:

For the [CLNT_UCAST_INST](#) request:

Windows implementations that use Microsoft Data Access Components (MDAC) or Windows Data Access Components (Windows DAC) time out if no response is received within 1 second. If a valid response is received within 1 second, the response is passed to the higher layer. If the response is not valid, the process is repeated.

Windows implementations that use SQL Server Native Client time out if no response is received within 1 second. If a valid response is received within 1 second, the response is immediately passed to the higher layer. If the response is not valid, an error is passed to the higher layer.

For the [CLNT_UCAST_DAC](#) request:

Windows implementations that use Microsoft Data Access Components (MDAC) or Windows Data Access Components (Windows DAC) do not support this request.

Windows implementations that use SQL Server Native Client time out if no response is received within 1 second. If a valid response is received within 1 second, the response is immediately passed to the higher layer. If the response is not valid, an error is passed to the higher layer.

<4> [Section 3.2.2](#): Windows implements the timers for these two messages as follows:

For the [CLNT_UCAST_EX](#) request:

Windows implementations that use Microsoft Data Access Components (MDAC) or Windows Data Access Components (Windows DAC) time out if no response is received within 0.5 second. If a valid response is received, it is appended to the results. If the response is not valid, it is discarded. The process is repeated until a time-out occurs.

Windows implementations that use SQL Server Native Client time out if no response is received within the lesser of 5 seconds or the specified logon time-out (the default logon time-out is 15 seconds.) If a valid response is received, it is appended to the results. If the response is not valid, it is discarded. The process is repeated for a maximum time period of the lesser of 5 seconds or the specified logon time-out.

For the [CLNT_BCAST_EX](#) request:

Windows implementations that use Microsoft Data Access Components (MDAC) or Windows Data Access Components (Windows DAC) time out if no response is received within 0.5 second. If a valid response is received, it is appended to the results. If the response is not valid, it is discarded. The process is repeated until a time-out occurs. There is no maximum time-out limit.

Windows implementations that use Microsoft SQL Server Native Client time out if no response is received within 5 seconds and then each 1 second up to 15 seconds or to the specified logon time-out, if valid responses are not received within each respective interval. If valid responses are received, they are appended to the results; however, invalid responses are discarded. The default logon time-out is 15 seconds.

<5> [Section 3.2.5.4](#): SQL Server Resolution Protocol clients consider a SVR_RESP message to a CLNT_UCAST_EX type request to be improperly formatted if the RESP_DATA field is more than 4,096 bytes.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

A

Abstract data model
[client](#) 19
[server](#) 17
[Applicability](#) 10

C

[Capability negotiation](#) 10
[Change tracking](#) 27
Client
[abstract data model](#) 19
[higher-layer triggered events](#) 20
[initialization](#) 20
[local events](#) 21
[message processing](#) 20
[overview](#) 18
[sequencing rules](#) 20
[timer events](#) 21
[timers](#) 19
[CLNT_BCAST_EX_packet](#) 11
[CLNT_UCAST_DAC](#) 23
[CLNT_UCAST_DAC_packet](#) 12
[CLNT_UCAST_EX](#) 22
[CLNT_UCAST_EX_packet](#) 11
[CLNT_UCAST_INST](#) 22
[CLNT_UCAST_INST_packet](#) 12

D

Data model - abstract
[client](#) 19
[server](#) 17

E

[Examples - overview](#) 22

F

[Fields - vendor-extensible](#) 10

G

[Glossary](#) 6

H

Higher-layer triggered events
[client](#) 20
[server](#) 17

I

[Implementer - security considerations](#) 24
[Index of security parameters](#) 24
[Informative references](#) 7
Initialization

[client](#) 20
[server](#) 17
[Introduction](#) 6

L

Local events
[client](#) 21
[server](#) 18

M

Message processing
[client](#) 20
[server](#) 17
Messages
[syntax](#) 11
[transport](#) 11

N

[Normative references](#) 7

O

[Overview \(synopsis\)](#) 8

P

[Parameters - security index](#) 24
[Preconditions](#) 9
[Prerequisites](#) 9
[Product behavior](#) 25

R

References
[informative](#) 7
[normative](#) 7
[Relationship to other protocols](#) 9

S

Security
[implementer considerations](#) 24
[parameter index](#) 24
Sequencing rules
[client](#) 20
[server](#) 17
Server
[abstract data model](#) 17
[higher-layer triggered events](#) 17
[initialization](#) 17
[local events](#) 18
[message processing](#) 17
[overview](#) 17
[sequencing rules](#) 17
[timer events](#) 18
[timers](#) 17

[Standards assignments](#) 10
[SVR_RESP packet](#) 13
[SVR_RESP_DAC packet](#) 15
[Syntax](#) 11

T

Timer events

[client](#) 21
[server](#) 18

Timers

[client](#) 19
[server](#) 17

[Tracking changes](#) 27

[Transport](#) 11

Triggered events - higher-layer

[client](#) 20
[server](#) 17

V

[Vendor-extensible fields](#) 10

[Versioning](#) 10